

Generating the ACME-project Cosmic Ray Shower Database

**Grant MacEwan University
Physical Science Department**

**Special Projects Course: PHYS 498
Project Supervisor: Dr. Orla Aaquist**

Tristan Lebel
Submitted: April 19, 2012

Generating the FUTURA Cosmic Ray Shower Database

The FUTURA project aims to design, prototype, and deploy an array of cosmic array detectors at the ATLAS site in CERN in order to analyze incoming cosmic ray showers. This detector array and the ATLAS detector need to be calibrated to determine the specific response characteristic of different types of air showers. Since cosmic ray air showers vary in angle and intensity and because they may have iron, protons, or iron as primary particles, each shower is unique. In order to test various theoretical configurations and shower events, a database of cosmic ray air shower simulations is needed. The database needs to contain simulations of showers having a variety of primary particles at several different energy levels and in a quantity large enough for analyses to be statistically significant. Therefore we need to generate potentially thousands of showers with data at both ground level at roughly 100 meters underground at the ATLAS detector.

The software package CORSIKA (Cosmic Ray Simulations for Kascade 6.990) is used to simulate cosmic ray showers. This software uses particle interaction models with a monte carlo simulation method to project cosmic ray particles through the atmosphere to ground level. The software runs in a linux or unix environment from the terminal window; and requires each shower to be launched in this manner, a process which is excessively time consuming. In order to save time and improve efficiency, a system was developed to automate this process, thereby taking full advantage of the capabilities of multi core CPUs .

Generating the database takes a significant amount of time since shower simulation times range from a few seconds for low energies to weeks at the upper end of the test range. Run time roughly scales with energy; however, due to randomness in monte carlo simulations, actual time will vary. Once generated, the data is binary encoded in named output files and corsikaread and another custom script (BATCO) are used to extract the data into an analyzable form that incorporates ground level energy distributions.

The final task is to simulate the transmission of cosmic rays underground to the ATLAS detector. It is well known and experimentally proven that cosmic rays quickly lose energy underground through various processes which are dependent both on the distance of strata traversed and the energy of the particle. There exist options for using software to do this task, however these are complex and more suited to small-scale work in biology. Instead, an original system was developed to track particles travelling to ATLAS through the geologic strata and underground structures.

A general knowledge of physics, linux operating systems, python programming, and geology is required to effectively use the software and the FUTURA-MacEwan Cosmic Ray Shower Database.

1. Configuring and Installing Corsika 6.990 for FUTURA Shower Database Generation

Corsika is a software package designed to simulate cosmic ray air showers in Earth's atmosphere. The software contains several packages which serve a variety of functions but for our experiments we are interested in the shower generation and data extraction software only. For the FUTURA Database we use the most current version of Corsika which is 6.990 as of April 13, 2012. For more information on the available features of Corsika, consult the instruction manual and website (Tanguy et. al. 2012).

Corsika 6.99x and other versions are coded in FORTRAN and are designed to be compiled and run in a Unix or Linux environment. Corsika has several software requirements that need to be installed: FORTRAN and c/c++ compilers are needed; lib2png may also be required. Since the output from Corsika can be up to several gigabytes per shower, sufficient storage must be made available. The Corsika documentation also recommends a minimum 1Ghz processor and 1024mb of memory (Tanguy et. al. 2012).

Two computers were used for the FUTURA simulations. The first machine is a Dell desktop with an Intel i7 2.8ghz 8 core processor, 8 gigabytes of RAM, and several hundred gigabytes of storage. This machine is running the Ubuntu 11.10 Linux operating system. The second test machine is an AMD Phenom 3.6Ghz 4 core processor

with 16 gigabytes of ram and sufficient storage. This system runs the Ubuntu 10.04 LTS Linux operating system. The Intel i7 system will be henceforth referred to as *pracom* and the AMD Phenom as *galileo*.

On both the *pracom* and *galileo*, some preparation is needed to install Corsika. This preparation begins with obtaining the Corsika software package. Corsika is freely available to anyone but is only downloadable through a secure server. A client computer host name and ip address must be submitted and are given permission to download; this can be done by following the instructions on the Corsika web site (Tanguy et. al. 2012). When we downloaded the software in December 2011, users were required to submit the IP address and host name (i.e. *pracom* / *galileo*). Once permission to download is given, the instructions for downloading supplied in the user's manual for Corsika can be followed. If this fails, using the *mget* Unix command may work. Information on *mget* can be found in the Unix man pages and may vary depending on the system and network used.

Once the Corsika 6.99x tar.gz package is downloaded, *pracom* and *Galileo* need to be prepared for installation. This involves installing two packages into the Ubuntu 10.04 and 11.10 operating systems using Ubuntu's Synaptic package manager. The first package is *gfortran*, the necessary Fortran 95 compiler; the second package, *lib2png*, which may not be necessary. In order to install these packages, root (sudo) privileges are needed (Canonical 2012).

Once the installation requirements are satisfied, installing Corsika is rather quick and easy. The first step is to create an install folder to which Corsika will be installed : */home/Corsika/*, for example. Once this folder is created, the installation files from the Corsika compressed file are extracted to this folder. The next step is to open a terminal session and 'cd' to the install folder that was created and then execute the *coconut* script. This will bring up a series of options for the install (we used all of the default selections). In addition, we installed the optional extra *plotsh2* (option 72), which required *lib2png*. *Plotsh2* is not necessary for the shower generation we conducted and was used only for testing. More information on *plotsh2* and other details are provided in the Corsika manual, which also provides a detailed overview of the installation process and contains solutions to errors that may occur.

2. Running Corsika and the Multicore Scripts.

2.1. Running Corsika – Normal Operations

Corsika, like many Unix packages, has no GUI. Because of this lack, Corsika must be launched from the command line and inputs are taken from *standard input* (Tanguy et. al. 2012). To launch an instance of Corsika, start a terminal session and 'cd' to the install folder then */run/* folder. Next launch the executable named *corsika6990Linux_QGSJET_gheisha*. This executable name may differ slightly on other installs.

Once running, Corsika will take configuration inputs from *standard input* one line at a time. Alternatively, configuration inputs can be passed from a text file using a PIPE command seen in the following Corsika launch command:

```
./corsika6990Linux_QGSJET_gheisha < INPUTFILE
```

Below is an example of a Corsika input file:

EVTNR	1	#Shower number, increment for each shower
NSHOW	1	#Showers to simulate (we do one at a time)
PRMPAR	14	#Particle type 14 = hadron, see Corsika manual
ESLOPE	-2.7	#Energy slope (Default value)
ERANGE	1.E5 1.E5	#Energy in GeV, min and max for shower
THETAP	0. 0.	#
PHIP	-180. 180.	#Shower direction (Default)

OBSLEV	110.E2	#Observation altitude in cm (Default)
FIXCHI	0.	#Default value, not used by us.
MAGNET	20.0 42.8	#Magnetic field center (Europe)
HADFLG	0 0 0 0 0 2	#Hadron interaction flags (Default)
ECUTS	0.05 0.02 0.02 0.02	#Energy cuts in GeV (Minimum values allowed)
MUMULT	T	#Hadron interaction stuff (Default)
ELMFLG	T T	#EM flags (Default)
STEPFC	1.0	#
RADNKG	200.E2	#
ARRANG	0.	#
LONGI	T 20. T T	#
ECTMAP	1.E3	#
MAXPRT	100	#Max particles to display (Default)
DIRECT	/home/tristan/1E5GeV-proton/	#Output folder
DATBAS	F	#Write dbase file
PAROUT	T F	#Particle output options (Default)
USER	FUTURA	#User
DEBUG	F 6 F 1000000	#Debug logs (Default)
PLOTSH	F	#Plotsh2 flag, needed if plotsh2 is installed
EXIT		#Ends the input

There are several more optional inputs, as well as settings for these inputs. The proper use of inputs and a list of all inputs can be found in the Corsika user manual.

Once the inputs have been passed to Corsika, the simulation is started. Depending on the energy given for the primary particle and the point of the first interaction, the shower may take from several seconds to weeks to run. This run time roughly scales with energy and is dependent on processor speed. Several files are generated by Corsika in the specified output folder; these are outlined below. Files are named by Corsika based on the given Event number.

a. DAT0000n

This file is a binary encoded file that contains the particle data. The file cannot be directly read using a text editor and must be processed using the binary executable *corsikaread* located under */run/* folder in the Corsika install folder. *Corsikaread* outputs a file called *fort.8* which is the binary file encoded as ASCII. Instructions for the manual operation of *Corsikaread* can be found in the user manual. For the FUTURA database, the binary files were analyzed using a different method that is covered in section 3.

b. DAT0000n.long

This file contains particle information encoded as long decimal values. This file is automatically generated, but we do not use or need it for the FUTURA Database.

Each set of files represents a cosmic ray air shower simulation. The FUTURA database requires thousands of simulations at various energies. This requirement presents two problems which need to be overcome. The first problem is that performing this many simulations manually would require countless hours of repeatedly formatting the input files and starting Corsika from the command line in addition to the time required to run the simulations themselves. The second problem is that Corsika 6.990 and older versions of Corsika do not support multi core processors, meaning that an instance of Corsika will use only one of the available processors, while the rest of the available computing power is wasted. This limitation can be overcome by running multiple instances of Corsika at the same time, one instance per core. Generating input files and starting Corsika can be done using scripting, which is where the Multicore Script comes in.

2.2. The Multicore Script

The Multicore Script was written in order to overcome the problems encountered when launching Corsika

(mentioned above in section 2.1) and take advantage of multiple processing cores. The script is written in python, therefore a UNIX/LINUX python interpreter and compiler are necessary to run it; Ubuntu 10.10 and 11.04 both have this natively and no install was necessary. The multicore script allows the user to generate a set number of showers at a given energy. For example, 1000 showers at 10^5 GeV, or 100 showers at 10^7 GeV. The output from Corsika is directed by the script to a specified output folder. The script also takes inputs based on energy, particle type, and number of showers to be generated, the details of which are covered later in this document. The script has two basic functions, the first being to generate inputs and launch Corsika, and the second being to schedule multiple instances of Corsika to run on multiple cores.

The input file generation is a simple process and is done using the following three steps:

- i. Using specified inputs and fixed values, specify the required Corsika input flags.

The Multicore script takes many ARGV type inputs from the comma line, allowing the user to specify the following quantities for Corsika: number of showers, particle energy cuts, particle type, and primary energy of the shower. There are many fixed values which are not supplied by the user and are meant to be kept constant for the FUTURA shower database. The input flags are generated with these fixed and supplied values.

- ii. From the generated input flags, create an ASCII formatted input file stored as a string.

Corsika requires a specifically-formatted input file where each line represents an input flag followed by the value and an optional comment denoted with the '#' symbol. An input file example is provided above in section 2.1. The Multicore Script uses the in-script-generated input flags and ASCII formatting to create a string with all the input flags separated by the new line character '\n'.

- iii. Pass this input file to Corsika through a PIPE and standard input.

When the input file is completed, an instance of Corsika is started using the *subprocess* libraries in python. Once started, the input file string is sent to Corsika using *subprocess.communicate(string)*; this may take a moment to complete. It has been found that if sufficient time is not given for the standard input PIPE to be cleared before starting the next instance of Corsika, corruption of the input can occur which will crash Corsika and the Multicore Script.

Once the first instance has begun, the input file for the next instance is generated and is ready to be passed to Corsika when the previous instance is finished. Running multiple instances at the same time is done in the following manner and adds a degree of complexity to scheduling:

- i. Start Corsika n times, once per allowed core.

Corsika is started using the *subprocess* library from python as mentioned previously. This allows the script to complete other tasks while Corsika runs and enables it to control the thread where Corsika is running. Corsika is launched in this way once per allowed core. The number of allowed cores is specified to the Multicore Script by ARGV input. An idle period of three seconds is given between each Corsika launch to allow for the standard input pipes to clear.

- ii. Once one instance has completed, start a new one and repeat this process until all showers have been run.

Once Corsika is running on all available cores, the script waits for an instance to terminate and checks

each thread every second to see if it is still running. The order in which the instance run by Corsika finish is random and so each thread must be monitored constantly. When one thread terminates, a new one is launched with a new input file. This is done until the specified number of runs has been completed.

- iii. Wait for all showers to terminate.

Once the final instance of Corsika has begun, the Multicore Script will wait for every instance to stop and then terminate. A report is generated which summarizes the inputs used and gives a sample input file and a report of how long it took to run Corsika.

2.3 Using the Multicore Script

The multicore script is launched in much the same way as Corsika in that a terminal sessions command line is used. However, unlike Corsika, the Multicore Script does not use standard input. Instead, ARGV values are used. This makes it easier for the Multicore Script to be launched using the shell script batch files used to generate the database. Below are the ARGV input flags as well as example launch shell commands.

<code>--inputFile <FILE></code>	Using this ARGV input allows the user to specify an outside input file for Corsika. When using this, the RUNNR and EXIT flags must be omitted since these are still assigned by the Multicore Script.
<code>--runs <INT></code>	The number of showers to be generated using the Multicore Script. This is a required flag.
<code>--start <INT></code>	The run at which to start. This is used only if a portion of the necessary runs has been completed and the user wants to continue on this series. This input will likely never be used during normal operations but was necessary during testing and is required when resuming from a computer crash.
<code> --out <STR></code>	Output path for the Corsika data using the format of the following example: <code>/home/Tristan/CorsikaOutput/</code>
<code>--corsika <STR></code>	The name given to the Corsika binary executable. I.e: <code>corsika6990Linux_QGSJET_gheisha</code>
<code>--path <STR></code>	The full path where the Corsika executable is located. E.g. <code>/home/tristan/CORSIKA/corsika-6990/run/</code>
<code>--ECUTSh <DEC></code>	Energy cuts for Hadrons in GeV. The minimum and default value is 0.05 GeV if this flag is not used.
<code>--ECUTSm <DEC></code>	Energy cuts for Muons in GeV. The minimum and default value is 0.02 GeV if this flag is not used.
<code>--ECUTSe <DEC></code>	Energy cuts for Electrons in GeV. The minimum and default value is 0.02 GeV if this flag is not used.

--ECUTSg <DEC>	Energy cuts for Gammas in GeV. The minimum and default value is 0.02 GeV if this flag is not used.
--particle <INT>	Particle type. See Corsika user manual for specific particle type integer values.
--energy <STR>	The energy of the primary particle in GeV, formatted as 1.E6 for 10^6 GeV.

Launching the Multicore Script can be done using the following shell command from the folder containing `corsika_multicore.py`. The following example was used to generate test data at 10^6 GeV for iron as the primary particle:

```
python corsika_multicore.py --runs 102 --start 1 --core 3 --particle 5626 --ECUTSh 0.05 --ECUTSm 0.02 --ECUTSe 0.02 --ECUTSg 0.02 --energy 1.E6 --corsika corsika6990Linux_QGSJET_gheisha --path /home/tristan/CORSIKA/corsika-6990/run/ --out /home/tristan/1E6GeV-iron/
```

Placing launch commands of this form in a shell batch file allows for showers of several different energies, particles, and output destinations to be run. Using such commands to launch the multicore script, the FUTURA Database was generated from analyses of the showers shown below which were produced by the Multicore Script.

The following showers with protons, iron, and photons as primary particles were simulated at given energy levels using the Multicore Script. Showers above 10^7 GeV took several days to weeks to run and produced massive files of up to 20 gigabytes; therefore, the number of simulations run for each primary particle was 100 and 8 for showers of 10^8 GeV and 10^9 GeV, respectively.

Energy (GeV)	Showers
10^4	1000
10^5	1000
10^6	1000
10^7	1000
10^8	100
10^9	8

3. Data extraction – Batch Data Analysis Tool for Corsika 6.99x Output, BATCO

As mentioned in section 2.1, Corsika creates output files that are encoded in binary and can only be read by a computer. Included with the Corsika software package is a tool called *corsikaread* which is an executable binary program designed to extract data and convert it to ASCII. This tool is written in Fortran, is not user friendly, and is slow to run. The output from *corsikaread* is dumped to a file called *fort.8* which is located under the Corsika install folder. Operation instructions for *corsikaread* are found in the Corsika user manual along with instructions for interpreting the *fort.8* file. The FUTURA database was generated using a python script that was written to extract the binary data using *corsikaread*, filter out everything but selected particle types, and then dump the data into an output that contains particle vector data. The script also creates particle energy distributions for each analyzed shower and produces a distribution that summarizes all showers. This script is called Batch Data Analysis Tool for Corsika 6.99x Output or BATCO.

3.1 Scripting *corsikaread* to automatically decode shower data files.

Extracting all the data manually using *corsikaread* would take an unreasonable amount of time and so the same approach taken to generate the database is used here: using python, *corsikaread* is automatically scripted using python to analyze all output data in a given folder.

When using the multicore script, the data files created are numbered sequentially in the following manner: *DAT000001*, *DAT000002*, *DAT000003*, and so on. This sequence represents the order in which the showers were generated. A series of data files in a folder represents multiple showers at a given energy, as specified in the summary report created by the Multicore Script. The BATCO python script functions in much the same way as the Multicore Script in that it analyzes the data sequentially starting with the first specified file and ending with the highest specified file. The basic functioning is shown in pseudo-code below.

```
Open file DAT00000n
  Start corsikaread using subprocesses libraries in python
    Wait for corsikaread to finish (loop and pause while checking thread)
  Open fort.8 file and import to string
    Remove top 78 lines from file, particles start at line 79.
    Loop: Read 1 particle until EOF is reached.
      If particle is the desired type, push it to the array.
    Output array to file titled DAT00000n.vec – this file contains particle vector data.
    Place each particle in a bin based on energy and write bin counts to DAT00000n.bins
  Delete fort.8
Repeat above steps until all files are completed.
```

It should be noted that the BATCO script does not use multi-coring because *corsikaread* only outputs to fort.8 and that this property cannot be changed nor the output redirected; therefore, if multiple instances of *corsikaread* are run simultaneously, data corruption will occur. Because of this limitation, data extraction can take a significant amount of time.

3.2 Using the BATCO Script

The BATCO script is launched using the command line and takes all inputs from ARGV flags. Detailed below are the input flags for BATCO.

--path <PATH>	This is the location of the corsikaread script, this must be supplied for the script to work and must be the full path.
--corsika <PATH>	Same as path, this can be ignored.
--data <PATH>	The folder location for the Corsika data files to be analyzed and extracted from binary form.
--start <INT>	The file number to start at, eg DAT00000n where n is the given integer.
--stop <INT>	The file number to stop at, eg DAT00000n where n is the given integer.
--bNum <INT>	Maximum number of bins to output in the energy distribution file. This is automatically set if the flag is not used.

-bEng <FLOAT>	The energy in GeV for each bin in the energy distribution file. This is automatically set if the flag is not used
--muon	Extract muons.
--antimuon	Extract antimuons
--gamma	Extract gamma rays
--proton	Extract protons
--electron	Extract electrons
--positron	Extract positrons

A sample launch command is provided below to give an example of a normal start command. This can be used in a shell script to analyze several folders of Corsika data automatically.

```
python BATCO.py --start 1 --stop 102 --path /home/tristan/CORSIKA/corsika-6990/run/ --data /home/tristan/1E6GeV-iron/ --muon
```

3.3 Output data from Batch Data Analysis Tool for Corsika 6.99x Output

BATCO generates 3 types of files: 2 files for every shower and one summary energy distribution file. All files outputted are encoded in ASCII text to be human-readable and in order to make working with the data easier. This system of encoding can be disadvantageous in that viewing the data using software like Microsoft Excel or in a text editor can be slow due to the potentially large size of the files.

The first type of file created by BATCO is the vector file. This file contains basic vector information such as momentum, energy, position, and direction. The file is tab delimited with each column representing a different quantity :

Column 1:	Particle ID – The type of particle based on the ID given by Corsika Muons are represented by the ID 6. Other ID's can be found in the Corsika user manual.
Column 2:	Energy (GeV) – The particle energy
Column 3:	X Momentum (GeV/c) – Momentum in the x direction on a left-hand coordinate system where z in the down direction is positive.
Column 4:	Y Momentum (GeV/c) – Momentum in the y direction in on a left hand coordinate system where z in the down direction is positive.
Column 5:	Z Momentum (GeV/c) – Momentum in the z direction in on a left hand coordinate system where z in the down direction is positive.
Column 6:	Altitude in degrees down from the horizontal.

Column 7: Azimuth in degrees. Zero degrees starts on the x axis at three o'clock.

Column 8: X Position (m) – The particle's x coordinate.

Column 9: Y Position (m) – The particle's Y coordinate.

Column 10: Radius (m) – The distance from the particle to the center of the shower.

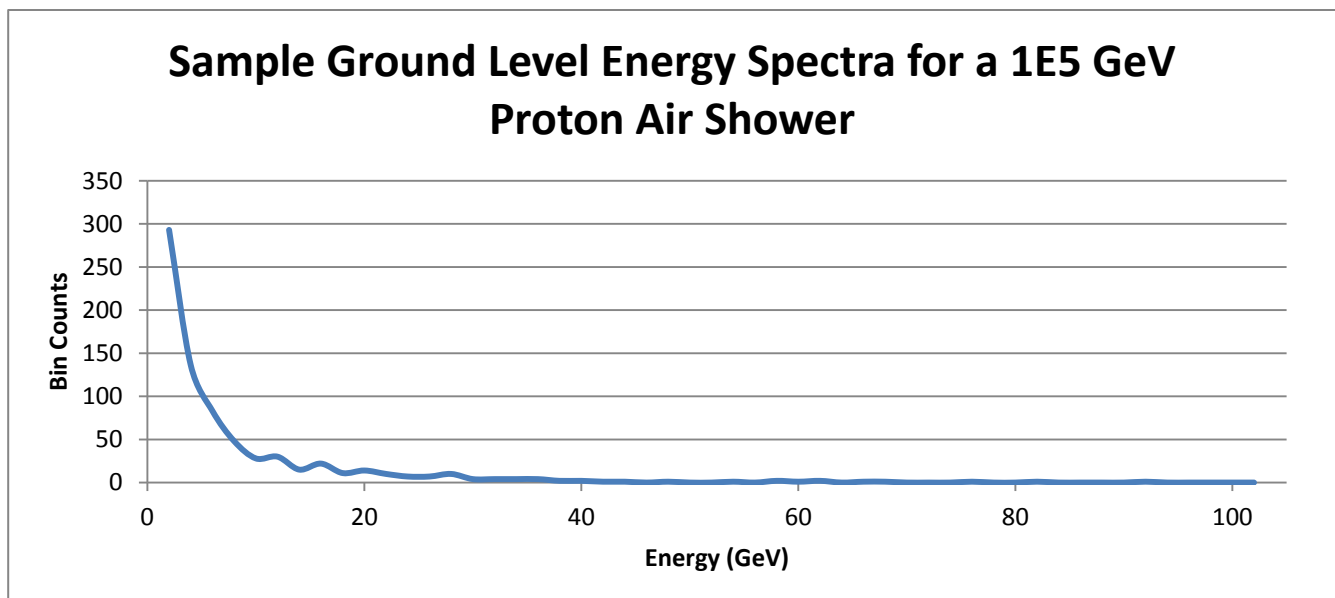
Contained within the DAT00000n<particle type>.vec files are all the necessary data to analyze the particles at ground level and underground during their transmission to the ATLAS detector. The remaining two types of files are energy distributions. These distributions are used only to examine the energy spectra of particles at ground level and do not contain particle data. The file is broken into three tab delimited columns, outlined as follows:

Column 1: Bin number. The maximum bin number can be set within the source code; however, at the moment, it is set automatically based on the highest energy particle in the data.

Column 2: Bin counts. The bin counts represent the total number of particles with energies between the energy of the previous bin and the energy of the current bin.

Column 3: Bin Energy. The maximum energy of the bin in GeV.

The two bin files, DAT00000n<particle type>.bins and Summary<particle type>.bins, contain the data needed to create energy distribution graphs that visualize the spectrum. Below is an energy spectrum from a 10^5 GeV proton air shower generated using Corsika with the Multicore Script and data extraction done by BATCO. The curve gives a general exponential decay shape which indicates that higher energy particles are less common than low energy particles.



4. Underground Particle Projections to ATLAS

Corsika only simulates cosmic ray air showers to ground level. To continue the projecting the air shower underground to the ATLAS site a separate software package is needed. This package has to take the output

vector files from BATCO and use this data to determine the number of particles to impact ATLAS as well as their energies. Rather than using and adapting a pre made software package to simulate particles underground it was decided that we would write our own. As such many things need to be considered.

The first is muon energy attenuation underground; high energy muons will pass through rock and lose energy at known rates. However, the rate of energy attenuation is a function of both distance of rock traversed and the initial energy of the muon. Using theoretical data and equations presented in Reichenbacher and De Jong 2007 a function for calculating energy attenuation was written. This function can be found under particle.py and is titled *lostEnergy(Energy,distance)*. The remaining physics was a matter of simple vector geometry to calculate such things as distance in three dimensions or if the particle lands within a geometric shape.

4.1 Methodology for Simulations

Outlined below is the methodology written in pseudo-code used to simulate underground particles. It should be noted that there are two main scripts used, the first formats the data and generates the output; the second analyzes a stack of particles through the underground structures of the ATLAS site and returns the particles which hit ATLAS to the first script. A third file of basic geometric and math functions is also used. Only the first script takes user input, the others are controlled from within. I was simpler and easier to work with multiple files rather than one big file.

Primary script particle_projector_3d.py:

```
Input all vector files from specified folder
  Read all particles from one vector file
  Append particles to stack
  Send stack be analyzed (separate script pEngine.py)
  Get returned stack of impact particles
  Output these particles along with average energy, shower size, shower location
  Loop: repeat for each file in the folder
Output showers analyzed, average particles, average impacts, average energies, shower location.
```

Particle analysis script pEngine.py:

```
Read input stack
Start processor pool (used for multi coring)
Pass stack to pool
  For each particle
    Step particle 5cm down in z direction
    Check to see if the particle is in a shaft, cavern or hit ATLAS
    Recorded distance traveled
    Record if this distance is in rock or in a shaft or cavern.
    Loop 5cm step until particle hits ATLAS or below maximum depth (~96m deep)
    If the particle hits ATLAS
      Calculate total distance from surface to impact location on ATLAS
      Use distance and ratio of rock vs air traversed to find total distance through rock.
      Apply energy loss based on rock distance and initial energy.
      Return impact particle to stack
  Return stack of impact particles
```

4.2 ATLAS Model

The model used of the underground ATLAS site is simplified to four basic geometric shapes. As a particle is tracked underground it is checked if the particle has interacted with these geometric shapes; this is the simplest way to determine what material the particle is passing through. Detailed below is the shapes and dimensions used.

- a. Vertical Shaft 1
 - Represented by a vertical cylinder
 - X Location = -19 meters from center of ATLAS
 - Y Location = 0 meters from center of ATLAS
 - Radius = 5 meters
 - Depth = 70 meters
- b. Vertical Shaft 2
 - Represented by a vertical cylinder
 - X Location = 17 meters from center of ATLAS
 - Y Location = 0 meters from center of ATLAS
 - Radius = 7 meters
 - Depth = 70 meters
- c. ATLAS Cavern
 - Represented by a rectangular prism
 - X Max = 25 meters, east side
 - X Min = -25 meters, west side
 - Y Max = 20 meters, north side
 - Y Min = -20 meters, south side
 - -Z Max = 65 meters depth, top of cavern
 - -Z Min = 100 meters depth, bottom of cavern
- d. ATLAS Detector
 - Represented by a horizontal cylinder
 - X Max = 22 meters, east side
 - X Min = -22 meters, west side
 - Y Location = 0
 - Radius = 12.5 meters
 - Depth of center = 83 meters

4.3 Using the scripts

Like with the multicore script and BATCO the underground simulator is launched using the command line and inputs are all passed via ARGV. Outlined below are the various input ARGV flags and a sample launch command.

--file <PATH>	Complete file path to a vector file including file name. This is not used as there has been no need for it yet.
--folder <PATH>	Path to folder containing vector files. Use this and not '--file <PATH>' ARGV input
--data <PATH>	The folder location for the Corsika data files to be analyzed and extracted from binary form.

--xOrg <INT> X location in meters of the shower origin. This allows for the shower to be offset to simulate a shower not centered above ATLAS. Default is 0 if the flag is not used

--yOrg <INT> Y location in meters of the shower origin. This allows for the shower to be offset to simulate a shower not centered above ATLAS. Default is 0 if the flag is not used

--rptNm <STRING> Name given to the report outputted once simulations are complete. The report name must be given in quotations, i.e. "Report Name". This flag is optional.

--energy <STRING> Energy of the air showers being analyzed. This flag is optional and is only used for generating the report.

4.4 Output

The underground particle projector is not yet completed as work on particle interactions within the ATLAS detector is continued. Because of this, the final output format has yet to be established, therefore all output data is written to Standard Output which can be piped in to a text file manually. Below is the output which is currently generated by the script; this format is not final and may include more information or different order in the future.

```
Primary particle energy:      1E5 GeV Proton
Number of showers analyzed:  8
Average particles per shower: 602 +/- 132
Average shower radius (m):   527 +/- 57
Average number of ATLAS impacts: 8 +/- 3
Average impact particle energy (GeV): 169 +/- 74
Shower center                x: 0 y: 0
```

Individual Shower Data:

```
=====
Shower:      1
Shower:      /media/BEE29397E293530F/CORSIKA_FILES/1E5GeV_proton/DAT000001-muon.vec
Total shower particles:      756
Mean shower radius (m):      455.952421492
Total impact particles:      8
Average impact particle energy (GeV): 116.023475766

Particle      Energy-GeV      Altitude      Azimuth      X (m) Y (m)
6      29.1454809633  89.7921796448  300.425423647  11.7169365463  -19.0521200697
6      143.080653331  89.9865291189  293.272602486  3.24527870537  0.0959068746771
6      21.7559146374  89.9396090272  179.338354965  1.76143340017  -7.32479799849
6      400.607636259  89.991922706  159.676910655  -1.19561396672  0.274462386361
6      45.1746051491  89.8646655453  70.8529791438  6.34851124027  11.0403957956
6      62.4217096928  89.9362630095  67.6688887923  4.82777716056  1.81228952235
6      206.922672498  89.968777665  255.451944296  0.216649212973  -7.078152241
6      19.0791335965  89.9012060151  279.011954412  0.64958378562  -10.9895935277
=====
```

Conclusions:

The Corsika software and accompanying python scripts have been developed to be used to help calibrate and simulate the ACME array being deployed by FUTURA at the ATLAS site. This software is still experimental and

work continues to be done on it specifically on the underground particle projection system.

Future work will include modeling the ATLAS detector to determine how many particles can pass through it and how many decay within or merely glance the detector. This will be included with the particle projector as a separate python script and will produce a list of particles.

As for Corsika, the version used to do generate the FUTURA database does not support multicore CPU's; however we developed a workaround. The next version of Corsika (7) should support multicore CPU's, this does not make our multicore script obsolete, however it will only be needed to launch Corsika and pass input and not manage multiple instance of it.

References:

Canonical Ltd (2012) Ubuntu Operating System 10.04LTS & 11.10 [Computer Software], Available from: <http://www.ubuntu.com>

Reichenbacher, J., De Jong, J., 2007, Calculation of the Underground Muon Intensity Crouch Curve from a Parameterization of the Flux at Surface, Available from: <http://arxiv.org/abs/0706.1110>

Tanguy, P., Heck, D., Knapp, J. (2012) Cosmic Ray Simulations for Kascade 6.990 (CORSIKA)[Computer Software], Das Institut für Kernphysik auf einen Blick, Retrieved December 2012. Available from: <http://www-ik.fzk.de/corsika/>

Thanks and Considerations

Grant MacEwan University

Research Supervisor:	Dr. Orla Aaquist
Particle Physics Mentor:	Dr. Evan Hackett
Student Peer Reviewer:	Lauren Huybregts

University of Alberta

FUTURA Project Leader:	Dr. James Pinfold
FUTURA Technical Consultant:	Dr. Richard Soluk